Abstract  It is shown how to apply the refinement calculus to stepwise refinement of both parallel programs and reactive programs. The approach is based on using the action systems model to describe parallel and reactive systems. Action systems are sequential programs which can be implemented in a parallel fashion. Hence the refinement calculus for sequential programs carries over to the parallel programs expressed in this framework. Refinement of reactive programs can be expressed and proved in the refinement calculus by using the methods of data refinement from the sequential refinement calculus.

Key words  Stepwise refinement, weakest preconditions, total correctness, parallel programs, reactive programs, refinement mappings, parallel composition, hiding, stuttering, action systems, fairness, simulation.

Contents

1. Introduction

2. Parallel programs as action systems
   2.1 Action system
   2.2 Parallel execution of action systems
   2.3 Fairness in action system executions
   2.4 Stepwise refinement of parallel programs

3. Refinement of action systems
   3.1 Iteration statements
   3.2 Data refinement
   3.3 Action systems
   3.4 Refinement with fairness constraints

4. Reactive programs as action systems
   4.1 Parallel composition and hiding
   4.2 Refinement of reactive components
   4.3 Permitting stuttering
   4.4 Example: Correctness of parallel execution model
   4.5 Refinement in context
5. Refinement of reactive systems
   5.1 Simulation refinement
   5.2 Preserving temporal properties

6. Example derivation
   6.1 Refinement by transformations
   6.2 Refining a reactive component

7. Concluding remarks

1 Introduction

In part I of this overview we have presented a lattice-theoretic framework for the refinement calculus. In this second part we apply this to the stepwise refinement of parallel and reactive programs. We will base our approach on the action system model for parallel programs.

The action system formalism for parallel and distributed computations was introduced by Back and Kurki-Suonio in [5] and is further developed in [8,7]. The behavior of parallel and distributed programs is described in terms of the actions that processes in the system carry out in co-operating with each other. Several actions can be executed in parallel, as long as the actions do not have any variables in common. The actions are atomic: if an action is chosen for execution, it is executed to completion without any interference from the other actions in the system.

Atomicity guarantees that a parallel execution of an action system gives the same results as a sequential and nondeterministic execution. We can therefore describe a parallel action system as a sequential statement in the language of commands. This allows us to use the sequential refinement calculus for stepwise refinement of action systems. We can start our derivation from a more or less sequential algorithm and successively increase the degree of parallelism in it, while preserving the correctness of the algorithm.

The refinement calculus is based on the assumption that the notion of correctness we want to preserve is total correctness. This is appropriate for parallel algorithms, i.e., programs that differ from sequential algorithms only in that they are executed in parallel, by co-operation of many processes. They are intended to terminate, and only the final results are of interest. Parallelism is introduced by merging action systems and refining the atomicity of actions. This approach to stepwise refinement of parallel algorithms has been put forward by Back and Sere [3,12].

In this paper we will concentrate on showing how the stepwise refinement method for action systems can be extended to also cover the stepwise refinement of reactive systems. Our starting point is the approach to refining reactive programs by refinement mappings put forward by Lamport in [25] and further developed by Abadi and Lamport [1], Stark [33,34], Jonsson [21], Lynch and Tuttle [27] and Lam and Shankar [24]. We will show that refinement of reactive systems can be seen as a special case of the general method for data refinement described in part I of this paper.

The action system approach is described in more detail in Section 2. Action systems will be just a special kind of block statements, consisting of an initialization and a loop. In Section 3 we show how to apply the general refinement theory to the specific case of refining action systems, with and without data refinement. In Section 4 we show how to describe reactive systems in this framework, and introduce operators for reactive composition. The notion of simulation refinement between reactive components is introduced based on data refinement, and we give a general method for refining reactive components of action systems. In Section 5 we study the notion of simulation refinement more closely and show that it can be used as such for refining reactive systems. Finally,